**Algorithms used and analysis of algorithm performance**

**Logistic Regression:**

**Processing of data:**

We created an empty Pandas data frame (design matrix) with headings, ‘SFH', 'PopUpWindow', 'SSL\_Final\_State', 'Request\_URL', 'URL\_of\_Anchor', 'Web\_Traffic', 'URL\_Length', 'Age\_Of\_Domain', 'IP\_Address' and finally 'Class' for each of the attribute columns. The data for each of these columns (which is pre-encoded) was then loaded in from a text file ‘data.txt’, split by a comma for each value within a line of text in the text file (see figure 3.2 for the populated design matrix). The data was then split into training, test and validation datasets which were further converted from panda data frames to NumPy arrays to implement analysis and operations on the respective data points and their values. With regards to splitting the data into training, test and validation sets, the Scikit class ‘sklearn.model\_selection’ and function ‘train\_test\_split’ were used giving a training dataset size of 811 and test and validation dataset sizes of 271 each.

(Citation: [Scikit-learn: Machine Learning in Python](http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html), Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.)

**Discussion, choice and implementation of algorithm:**

Due to the well-defined, highly discrete and categorical nature of our target variables, using logistic regression (with gradient descent and without regularization) was an easy choice since we’re using different web attributes to predict the legitimacy of a website, which implied the use probabilistic outputs to make our predictions. We found that logistic regression lends itself very well to multiclassification tasks. Therefore, we applied gradient descent (the next paragraph gives an in-depth explanation of the hyperparameter values and why) with the concept of the “1 vs all” property to obtain three binary models (to represent our 3 target variables). These 3 models had their own respective sets of theta values which simplified the classification process tremendously. Finally, the classification occurred through the SoftMax function which made very intuitive sense as it multiplied each data row through each set of theta values from the binary models, obtained probabilities and classified according to the highest of the three probabilities for each data row.

**Adjustment of Hyperparameters and Results:**

The 2 main hyperparameters used in this algorithm were the tolerance and learning rate. Tolerance is the value that the difference between the old and new must be greater than in order to continue the gradient descent. After doing a multitude of test runs using different values for the tolerance and learning rate, we concluded that we got the best possible performance and total test accuracy (least total test error) using the values of tolerance = 0.001 and learning rate = 0.001. While values greater than 0.001 for the tolerance and learning rate would also give very similar results, they would result in either model 2 and 3 having a low iteration count with fewer updates to the theta parameters which led to marginally lower test data accuracy and slightly higher errors. Finally, very small values (<0.001) of tolerance and learning rate, would result in the gradient descent looping almost infinitely, taking extremely long to update the theta parameters to their optimal values which was very inefficient and redundant. With the values of tolerance and learning rate = 0.001, we obtained very promising results and visuals. A training accuracy of 82%, a validation accuracy of 81% and a test accuracy of 80%. Furthermore, for binary models 1 and 2, the errors would decrease iteratively until they reached an extremely small value which was a promising observation. Model 3’s error would increase over time but very marginally which may have been the cause of the 18-20% misclassification rate/error (See figures 1.1,1.2,1.3 for the error plot under of tolerance = 0.001 and learning rate = 0.001.). The training, test and validation accuracies and errors are all displayed below along with a confusion matrix for the predictions on the test data where the learning rate = 0.001 and the tolerance = 0.001. (See figures 2.1,2.2,2.3 and 3.1). To conclude, if someone else were working with this data, I’d recommend that they try to minimise the errors across all 3 binary models as much as possible to get a higher accuracy, possibly by implementing regularization to this logistic regression algorithm.

**Error plot and result figures and confusion matrix under tolerance = 0.001 and learning rate = 0.001:**

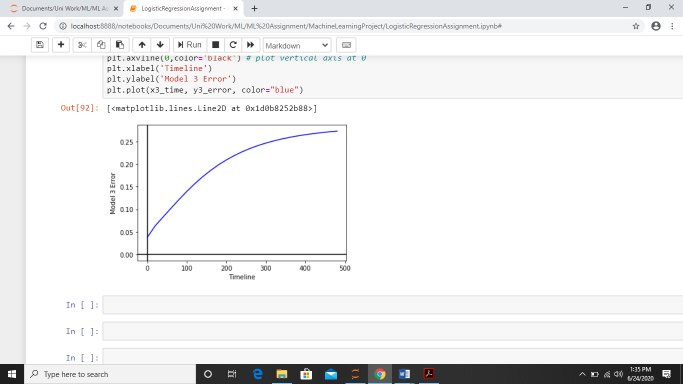
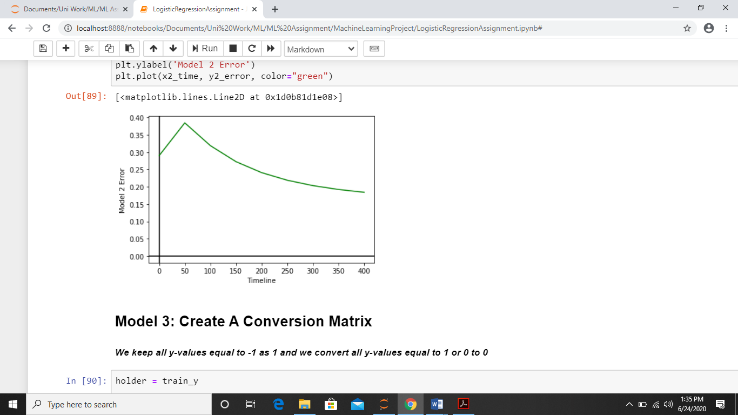
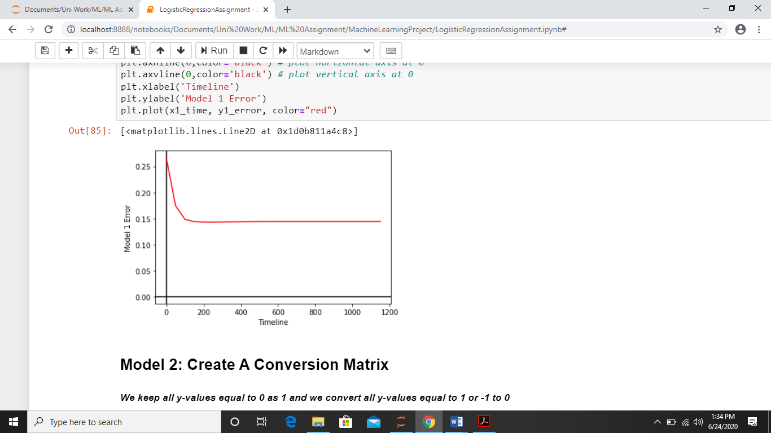


Figure 1.2

Figure 1.1

Figure 1.3

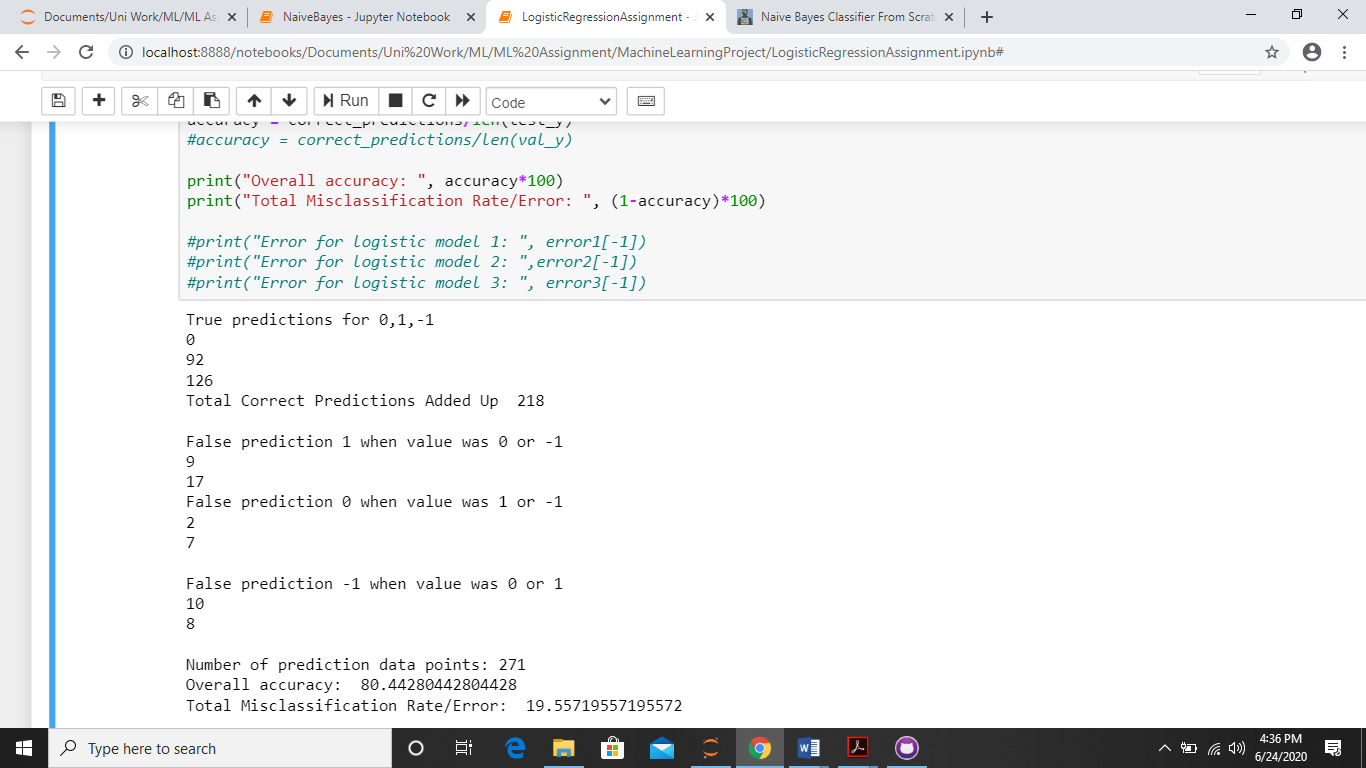
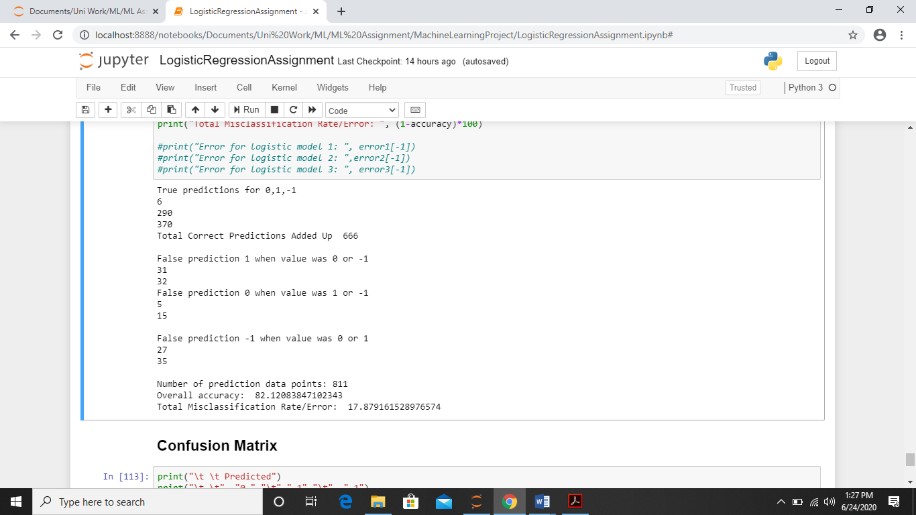
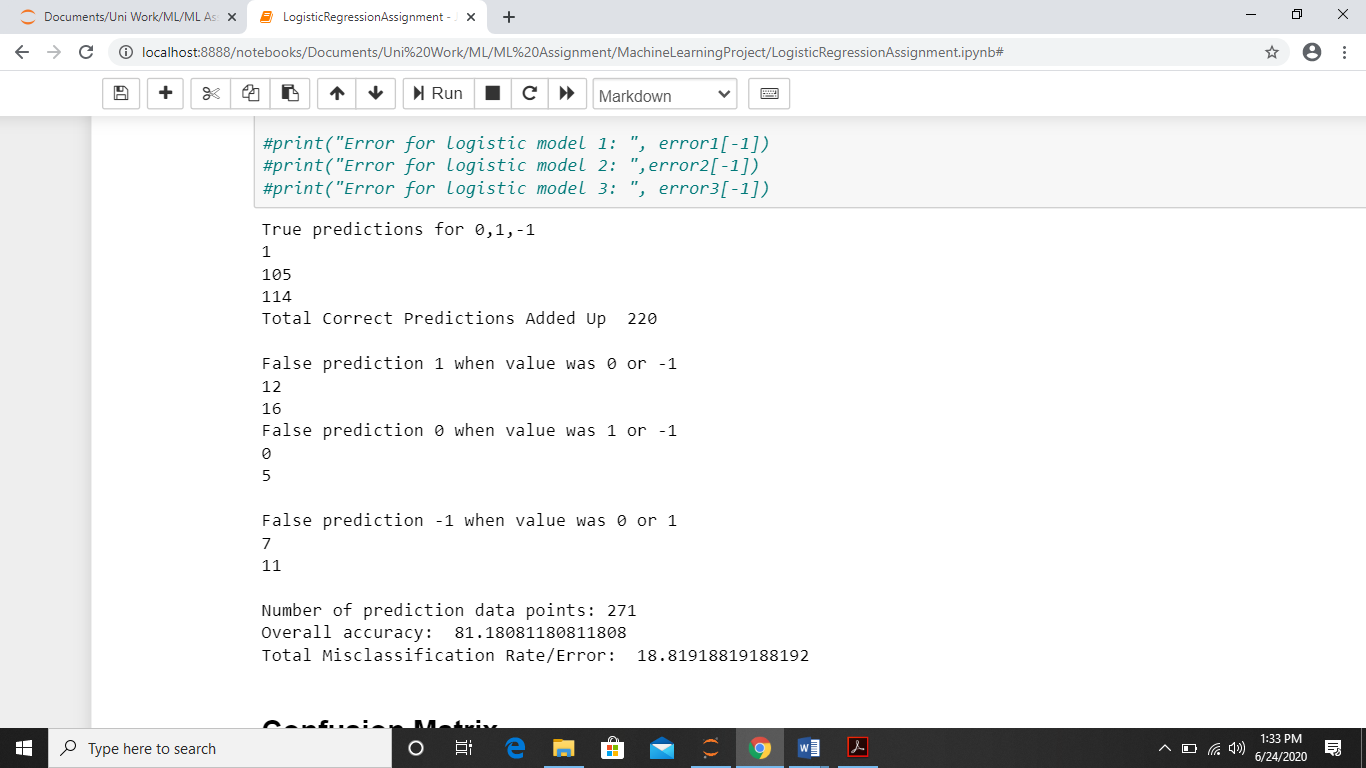


Figure 2.3 – Test accuracy, error and data analysis

Figure 2.2 – Validation accuracy, error and data analysis

Figure 2.1 – Training accuracy, error and data analysis

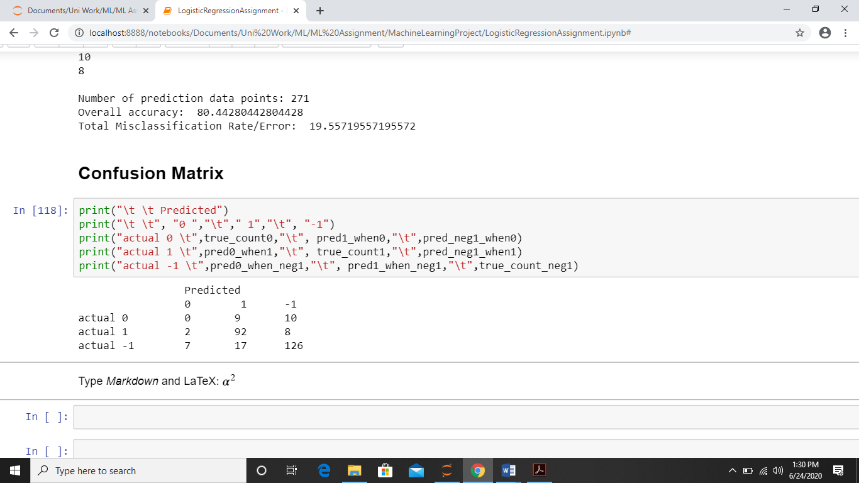
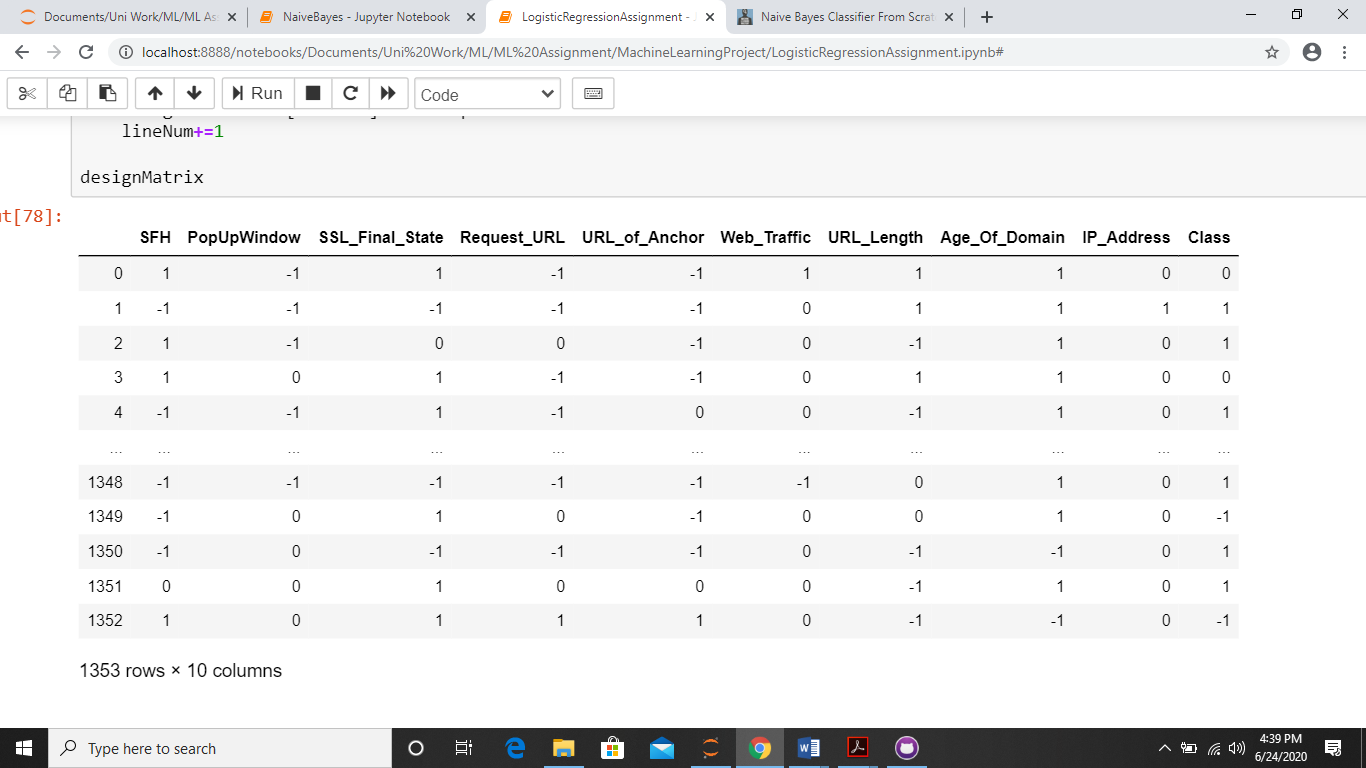


Figure 3.2 – Populated design matrix with data from text file

Figure 3.1 – Confusion matrix of test data predictions